ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ
INFORMATION TECHNOLOGY, COMPUTER SCIENCE, AND MANAGEMENT



УДК 519.683.4 DOI 10.12737/19687

Исследование влияния шаблона доступа к глобальной памяти графического процессора на производительность*

Р. В. Арзуманян[1], А. И. Сухинов[2]**

[1] Институт компьютерных технологий и информационной безопасности Южного федерального университета, г. Таганрог, Российская Федерация
[2] Донской государственный технический университет, г. Ростов-на-Дону, Российская Федерация

Study of access template to graphics engine GM effect on the performance ***

R. V. Arzumanyan[1], A. I. Sukhinov[2]**

[1] Institute of Computer Technology and Information Security, Southern Federal University, Taganrog, Russian Federation
[2] Don State Technical University, Rostov-on-Don, Russian Federation

Целью данной работы является изучение влияния загруженности вычислительных ядер графического процессора и шаблона обращения к памяти на пропускную способность шины памяти и ускорение масштабирования. Предмет исследования — задача масштабируемости производительности параллельных вычислений и их ускорение. В ходе работы была проверена следующая гипотеза: при обработке изображений для многоядерных систем с общей памятью закон Густафсона — Барсиса более важен, нежели шаблон обращения к памяти при недостаточной загруженности вычислительных ядер графического процессора. Методология проведения исследования — вычислительный эксперимент с последующим анализом полученных данных. В ходе исследования подтверждена выдвинутая гипотеза. Для этого был проведен ряд экспериментов на различных гетерогенных вычислительных системах с поддержкой стандарта OpenCL. Анализ результатов позволяет сделать следующие выводы. Шаблон доступа к памяти начинает накладывать ограничения на производительность алгоритма только при достаточной загруженности вычислительных ядер. Видеокарты, оснащенные собственной памятью, показывают более стабильные результаты работы по сравнению с теми, что используют память, общую с центральным процессором.

Область применения полученных данных — разработка алгоритмов и программного обеспечения для высокопараллельных вычислительных систем.

Ключевые слова: GPGPU, доступ к памяти, OpenCL.

The work objective is to study the effect of the graphical processor unit computational cores load level and memory access pattern on the memory bus bandwidth and scaling acceleration. The research subject is the problem of scalability of the parallel computing performance and acceleration. The following hypothesis is checked: while processing images for multi-core shared-memory systems, Gustafson – Barsis's law is more crucial than the memory access template at the underloading of the GPU cores. The research methodology is a computational experiment with further analysis of the obtained results. The conclusions are as follows. The suggested hypothesis is proved. For that, a series of experiments on various heterogeneous computational systems with OpenCL standard support is conducted. The application field of the results obtained includes the development of algorithms and software for the highly parallel computer systems. The memory access template starts to place certain restrictions on the algorithm efficiency only when the load level of the computational cores is sufficient. Video cards with the private memory show more stable results in comparison to those which share memory with the central processing unit.

Keywords: GPGPU, memory access template, OpenCL.

Введение. В настоящее время признано, что графические процессоры (graphical processing unit — GPU) являются мощным инструментом решения задач, хорошо поддающихся распараллеливанию (General-purpose computing for graphics processing units, GPGPU — неспециализированные вычисления на графических процессорах). Однако лишь малая часть существующего программного обеспечения эффективно использует мощности графических процессоров.

---

*Работа выполнена в рамках инициативной НИР.
**E-mail: roman.arzum@gmail.com, sukhinov@gmail.com
***The research is done within the frame of the independent R&D.

В качестве причин могут быть названы относительная новизна (широкое распространение технологии GPGPU получили в 2008–2010 гг.) и существенное архитектурное отличие от процессоров приложений (большее количество ядер, малый суммарный объем кэш-памяти).

Вопросу оптимизации алгоритмов для GPU посвящено немало руководств по программированию от ведущих производителей видеокарт. Среди них можно отметить документы от Nvidia [1], AMD [2] и ARM [3]. Во всех перечисленных источниках организация доступа потоков графического процессора к памяти представлена как важный аспект в достижении высокой производительности многоядерных систем. Однако в периодической научной литературе авторам не удалось найти работы, систематически исследующие, каким образом на производительность систем влияет шаблон доступа к памяти различных систем с графическими процессорами. Ранее влияние операций обмена информацией при обращении к памяти и при межпроцессорных пересылках рассматривались в работах [4–6].

В руководствах в общих чертах сказано о том, что доступ к памяти нужно организовывать определенным образом — так, чтобы соседние потоки обращались к соседним ячейкам памяти [1]. Желательно, чтобы при этом использовались векторные операции чтения/записи [2, 3]. На практике это не всегда возможно по ряду причин (шаблон доступа жестко задан алгоритмом, используется структура данных в формате изображения, которая не позволяет организовать такое обращение к оперативной памяти, и т. п.).

В этой связи интересно выяснить, на какой уровень производительности можно рассчитывать при обработке изображений с использованием различных шаблонов доступа, необходимых, чтобы определить, насколько «неудачный» шаблон может снизить скорость работы алгоритма. Также важно ответить на вопрос: является ли доступ к памяти тем фактором, который ограничивает производительность алгоритма в первую очередь? Помимо вышеперечисленного интерес представляет проведение экспериментов на компьютерах, в которых центральный и графический процессоры используют физически общую память, поскольку в данном случае возрастает нагрузка на общий контроллер памяти и результаты могут быть менее предсказуемыми.

Гипотеза. Как и для всякой вычислительной системы с массовым параллелизмом, для GPU важен закон Густафсона — Барсиса [7, 8]:

$$S_p = g + (1-g)p, \quad (1)$$

где g — доля последовательных расчетов, p — количество процессоров, S_p — ускорение масштабирования.

В качестве модельной выберем задачу копирования изображения с различным числом потоков. Обработка изображений является типичной задачей для GPU. Выдвинем гипотезу: «Количество задействованных потоков в вычислениях оказывает большее влияние на быстродействие алгоритма для GPU, нежели шаблон доступа к памяти при малом количестве потоков». Несмотря на кажущуюся очевидность данного утверждения, не следует забывать о том, что в случае неудачной организации доступа к памяти запросы могут быть сериализованы [1, 2], что приведет к увеличению g.

Цель работы. Предполагается ответить на ряд вопросов.

— Является ли доступ к памяти тем фактором, который ограничивает производительность алгоритмов на GPU в первую очередь?

— Существует ли шаблон доступа, одинаково подходящий для большинства GPU?

— Каков разброс результатов для видеокарт с собственной памятью и тех, что используют память из общего с CPU объема?

Экспериментальная часть. Каждый эксперимент представляет собой запуск OpenCL ядра, которое осуществляет копирование монохромного изображения 384×512 пикселов с 8-битным цветом. При этом используется уникальный шаблон доступа к памяти.

Ширина изображения в пикселах обозначалась W; высота изображения в пикселах — H, количество потоков по оси $0X$ — Thx, количество потоков по оси $0Y$ — Thy.

Всего использовалось 9 шаблонов доступа, что дает столько же экспериментов (табл. 1).

Таблица 1

Описание экспериментов

Шаблон	Размерность	Число потоков	Особенности копирования пикселов потоком
Simple	$Th_x=W$ $Th_y=H$	196608	Копирует один пиксел
Row4	$Th_x=W/4$ $Th_y=H$	49152	Векторно копирует строку из 4 пикселов
Row16	$Th_x=W/16$ $Th_y=H$	12228	Векторно копирует строку из 16 пикселов
Coll4	$Th_x=W$ $Th_y=H/4$	49512	Поэлементно копирует столбец из 4 пикселов
Col16	$Th_x=W$ $Th_y=H/16$	12228	Поэлементно копирует столбец из 16 пикселов
Row4×4	$Th_x=W/4$ $Th_y=H/4$	12228	В цикле векторно копирует строку из 4 пикселов
Row16×16	$Th_x=W/16$ $Th_y=H/16$	768	В цикле векторно копирует строку из 16 пикселов
Col4×4	$Th_x=W/4$ $Th_y=H/4$	12228	В цикле поэлементно копирует столбец из 4 пикселов
Col16×16	$Th_x=W/16$ $Th_y=H/16$	768	В цикле поэлементно копирует столбец из 16 пикселов

Эксперименты проводились на четырех тестовых компьютерах (табл. 2).

Таблица 2

Характеристики тестовых компьютеров

Система	CPU	GPU	Тип памяти GPU
Windows 8.1	Core i5 4200U	AMD Radeon 8670M	Дискретная, 1Gb DDR3 800 MHz, 128 bit
Windows 7	AMD A10-5700K		Общая с CPU, DDR3 900 MHz, 128 bit
Linux	ARM CortexA15	ARM Mali T624	Общая с CPU, DDR3 800 Mhz, 64 bit
Windows Server 2008	AMD Opteron 6100	Nvidia Tesla C2075	Дискретная, 6Gb DDR5 3600 MHz, 384 bit

Тестовые ПК можно разделить на две категории:

1) процессор и видеокарта используют общую память из объема ОЗУ;

2) видеокарта оснащена собственной памятью.

Это важно, потому что использование общей памяти создает дополнительную нагрузку на контроллер памяти.

Стандарт OpenCL позволяет использовать общую память и для видеокарт с дискретной памятью — в этом случае операции чтения/записи выполняются неявно для пользователя. Поэтому эксперименты проводились в двух режимах. Слева на графике (рис. 1) — вариант с использованием памяти видеокарты. Справа — вариант с использованием общей памяти. По оси абсцисс — название эксперимента.

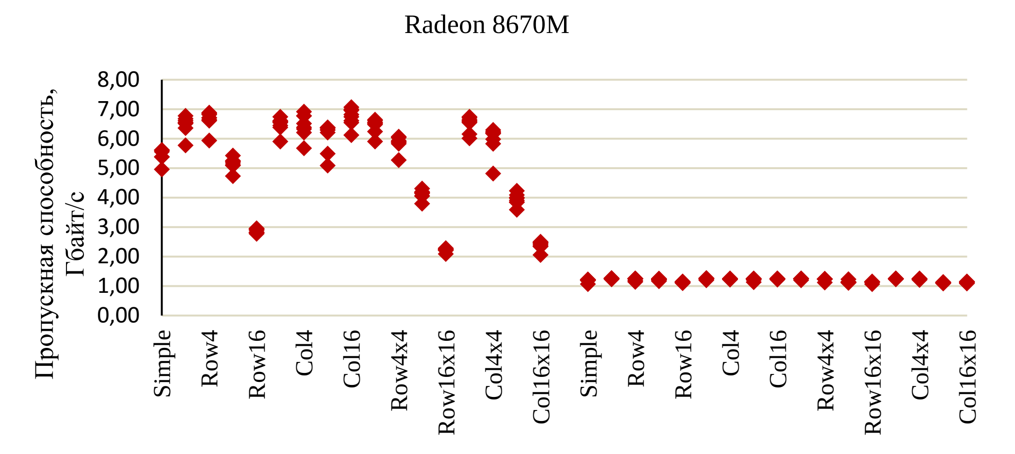


Рис. 1. Экспериментальные результаты для видеокарты AMD Radeon 8670M

Данная видеокарта основана на RISC-архитектуре GCN [8]. При использовании памяти видеокарты пропускная способность почти всегда высокая (около 50 % от пиковой) и всегда стабильная. При использовании разделяемой памяти ограничителем скорости является шина PCI-E 2.0 X4, к которой подключен GPU. Отчетливо видно, что при малом числе потоков (эксперименты Row16×16 и Col16×16) результаты схожи, несмотря на различный шаблон доступа. При этом результаты экспериментов с большим числом потоков (Row16 и Col16) различаются более чем в 2 раза — следовательно, шаблон доступа к памяти в данном случае имеет значение.

Карта Radeon 7660D основана на VLIW-архитектуре Terascale, предшествовавшей GCN. Несмотря на большую пиковую пропускную способность (21 Гбайт/с) по сравнению с предыдущим экспериментом, полученные результаты в среднем хуже и, что более важно, существенно менее стабильны. Ситуация с экспериментами Row16, Col16, Row16×16 и Col16×16 в точности повторяет результаты предыдущей видеокарты (рис. 2).

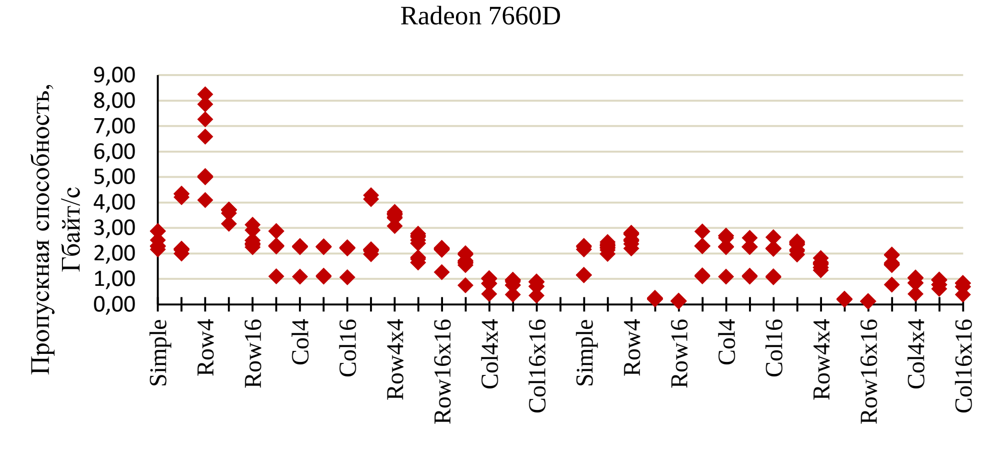


Рис. 2. Экспериментальные результаты для видеокарты AMD Radeon 7660D

Mali T624 — видеокарта для мобильных устройств [9]. Данный GPU основан на VLIW-архитектуре ARM Midgard. В данном случае видна сильная зависимость результата от длины векторных команд чтения/записи. С ростом длины команды пропускная способность также возрастает. В то же время поэлементный доступ к элементам изображения предсказуемо медленный. Примечательно, что использование собственной или разделяемой памяти никак не сказывается ни на пропускной способности, ни на разбросе результатов. В данном случае шаблон доступа к памяти имеет гораздо большее значение, чем число потоков. Это можно объяснить тем, что видеокарта оснащена всего 8 ALU, и даже относительно малое число потоков загружает графический процессор (рис. 3).

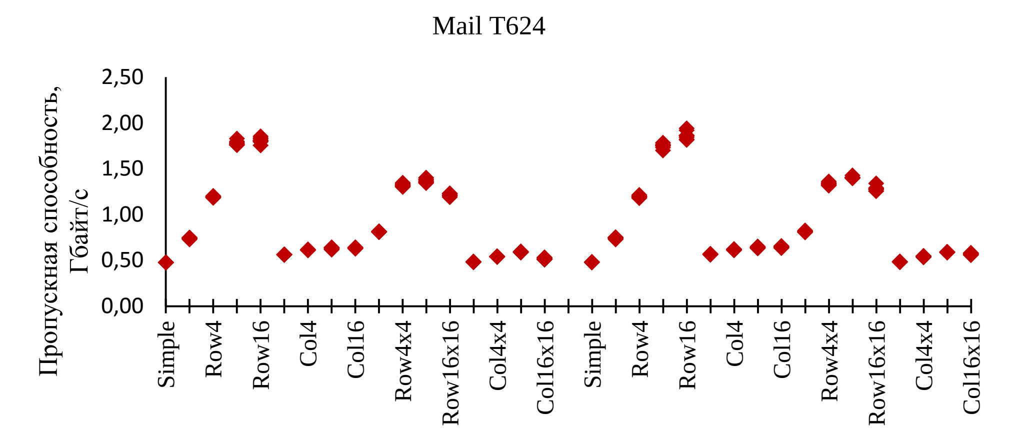


Рис. 3. Экспериментальные результаты для видеокарты ARM Mali T624

Tesla C2075 — это флагманский ускоритель вычислений Nvidia на архитектуре Fermi [10]. Данная карта оборудована 6 Гб собственной памяти. По результатам экспериментов можно сделать вывод о том, что объем задачи недостаточен для такого мощного решения, т. к. максимальная достигнутая пропускная способность составляет всего 12 % от пиковой. Использование собственной или разделяемой памяти никак не отразилось на результатах, потому что ускоритель подключен через шину PCI-E 2.0 X16, которая не является сдерживающим фактором. В данном эксперименте шаблон доступа к памяти также становится важен при большом числе потоков (эксперименты Row4 и Col4). При меньшем числе потоков (эксперименты Row16×16 и Col16×16) шаблон доступа оказывает не такое большое влияние на производительность (рис. 4).

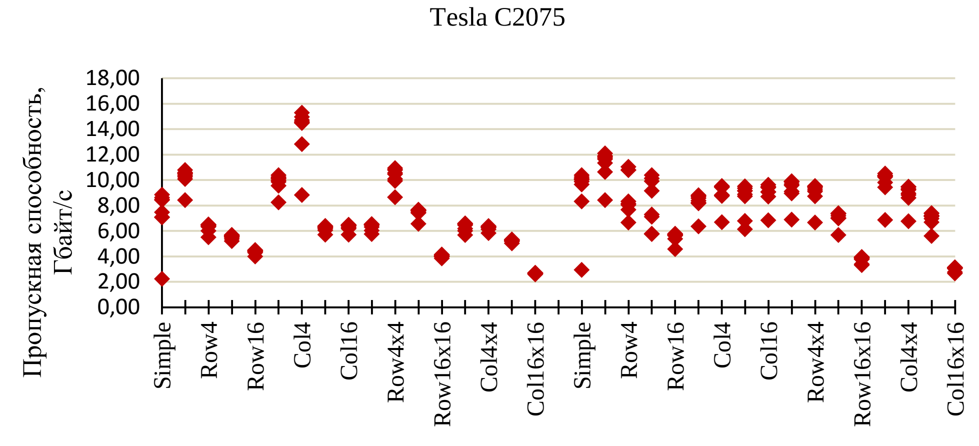


Рис. 4. Экспериментальные результаты для видеокарты Nvidia Tesla C2075

Выводы. По результатам проведенных экспериментов можно сделать вывод о том, что для тестовых систем высказанная гипотеза является справедливой — а именно, шаблон доступа к памяти начинает накладывать ограничения на производительность алгоритма только при достаточно большом количестве запущенных потоков.

В ходе экспериментов не найден метод, который показывал бы одинаково хороший результат на всех протестированных системах, несмотря даже на малое их количество.

Видеокарты, оснащенные собственной памятью, предсказуемо показывают более стабильные результаты работы, но при этом накладывают ограничения на совместное использование данных центральным и графическим процессорами. Напротив, видеокарты, использующие общую с центральным процессором память, могут сохранять стабильность результатов при использовании общей памяти, а также давать больший разброс показателей.

Библиографический список

1. OpenCL Programming Guide for the CUDA Architecture / NVIDIA corporation. — Santa Clara : NVIDIA, 2009. — 60 p.
2. AMD Accelerated Parallel Processing OpenCL Programming Guide / Advanced Micro Devices. — Sunnyvale : ADM, 2013. — 294 p.
3. Mali T600 Series OpenCL GPU Developer Guide [Электронный ресурс] / ARM. — Режим доступа: http://infocenter.arm.com/help/topic/com.arm.doc.dui0538e/DUI0538E_mali_t600_opencl_dg.pdf (дата обращения: 16.04.16).
4. Сухинов, А. И. Двумерные схемы расщепления и некоторые их приложения / А. И. Сухинов // Москва : МАКС Пресс, 2005. — 408 с.
5. Николаев, И. А. О распараллеливании треугольных итерационных методов на специализированной многопроцессорной системе / И. А. Николаев, А. И. Сухинов, О. Д. Харина // Автоматика и телемеханика. — 1986. — Вып. 5. — С. 135–142.
6. Сухинов, А. И. Локально-двумерные схемы для решения многомерных параболических уравнений на вычислительных системах матричного типа / А. И. Сухинов // Известия вузов. Математика. — 1984. — № 11. — С. 45–53.
7. Encyclopedia of Parallel Computing / Ed. D. Padua. — New York : Springer, 2011. — 2176 p.
8. Quinn, M.-J. Parallel Programming in C with MPI and OpenMP / M.-J. Quinn. — New York : McGraw-Hill, 2003. — 516 p.
9. AMD Graphic Core Next [Электронный ресурс] / Advanced Micro Devices // AMD Fusion Developer Summit 2013. — Режим доступа: http://developer.amd.com/wordpress/media/2013/06/2620_final.pdf (дата обращения: 16.04.16).
10. Global Internet Phenomena Report [Электронный ресурс] / Sandvine. — Режим доступа : https://www.sandvine.com/trends/global-internet-phenomena/ (дата обращения: 16.04.16).

References

1. OpenCL Programming Guide for the CUDA Architecture. NVIDIA corporation. Santa Clara: NVIDIA, 200, 60 p.
2. AMD Accelerated Parallel Processing OpenCL Programming Guide. Advanced Micro Devices. Sunnyvale: ADM, 2013, 294 p.
3. Mali T600 Series OpenCL GPU Developer Guide. ARM. Available at: http://infocenter.arm.com/help/topic/com.arm.doc.dui0538e/DUI0538E_mali_t600_opencl_dg.pdf (accessed: 16.04.16).
4. Sukhinov, A.I. Dvumernye skhemy rasshchepleniya i nekotorye ikh prilozheniya. [Two-dimensional splitting schemes and some of their applications.] Moscow: MAKS Press, 2005, 408 p. (in Russian).
5. Nikolayev, I.A., Sukhinov, A.I., Kharina, O.D. O rasparallelivanii treugol'nykh iteratsionnykh metodov na spetsializirovannoy mnogoprotsessornoy sisteme. [On parallel application of triangular iterative methods in a special-purpose multiprocessor system.] Avtomatika i Telemekhanika, 1986, iss. 5, pp. 135–142 (in Russian).
6. Sukhinov, A.I. Lokal'no-dvumernye skhemy dlya resheniya mnogomernykh parabolicheskikh uravneniy na vychislitel'nykh sistemakh matrichnogo tipa. [Locally two-dimensional schemes for solving multidimensional parabolic equations in computer systems of matrix type.] Izvestiya VUZ. Matematika, 1984, no. 11, pp. 45–53 (in Russian).
7. Padua, D., ed. Encyclopedia of Parallel Computing. New York: Springer, 2011, 2176 p.
8. Quinn, M.-J. Parallel Programming in C with MPI and OpenMP. New York: McGraw-Hill, 2003, 516 p.
9. AMD Graphic Core Next. Advanced Micro Devices. AMD Fusion Developer Summit 2013. Available at: http://developer.amd.com/wordpress/media/2013/06/2620_final.pdf (accessed 16.04.16).
10. Global Internet Phenomena Report. Sandvine. Available at: https://www.sandvine.com/trends/global-internet-phenomena/ (accessed: 16.04.16).

Поступила в редакцию 29.01.2016
Сдана в редакцию 29.02.2016
Запланирована в номер 23.03.2016